МІНІСТЕРСТВО ОСВІТИ І НАУКИ КРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського”
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

МЕТОДИ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНИХ МЕХАНІЗМІВ
ЛАБОРАТОРНА РОБОТА №1

“Вибір та реалізація базових фреймворків та бібліотек ”

Виконали:
студенти 6-го курсу
групи: ФІ-31мн
Коробан Ольга
Каюк Ксенія
Кухар Богдан

Київ - 2024

Мета лабораторної роботи:

Порівняння бібліотек OpenSSL, Crypto++, CryptoLib, PyCrypto для розробки гібрідної криптосистеми під Android/MacOs/IOs платформу

OpenSSL

Встановлення OpenSSL використовуючи менеджер пакетів Homebrew

```
openssl@3 3.3.1 is already installed but outdated (so it will be upgraded).
==> Downloading https://ghcr.io/v2/homebrew/core/openssl/3/manifests/3.4.0
######################################################################## 100.0%
==> Fetching dependencies for openssl@3: ca-certificates
==> Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/manifests/2024-
######################################################################## 100.0%
==> Fetching ca-certificates
==> Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/blobs/sha256:21
######################################################################## 100.0%
==> Fetching openssl@3
==> Downloading https://ghcr.io/v2/homebrew/core/openssl/3/blobs/sha256:bf2e6c5c
######################################################################## 100.0%
==> Upgrading openssl
  3.3.1 -> 3.4.0
==> Installing dependencies for openssl@3: ca-certificates
==> Installing openssl@3 dependency: ca-certificates
==> Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/manifests/2024-
Already downloaded: /Users/a1/Library/Caches/Homebrew/downloads/338dad7c2ff7c822cda7c417944521589856741c0fbd7a7f07b8
8a18d7fb7e05--ca-certificates-2024-09-24.bottle_manifest.json
==> Pouring ca-certificates--2024-09-24.all.bottle.tar.gz
==> Regenerating CA certificate bundle from keychain, this may take a while...
🍺  /opt/homebrew/Cellar/ca-certificates/2024-09-24: 4 files, 237.4KB
==> Installing openssl@3
==> Pouring openssl@3--3.4.0.arm64_sequoia.bottle.tar.gz
==> Caveats
A CA file has been bootstrapped using certificates from the system
keychain. To add additional certificates, place .pem files in
  /opt/homebrew/etc/openssl@3/certs

and run
  /opt/homebrew/opt/openssl@3/bin/c_rehash
==> Summary
🍺  /opt/homebrew/Cellar/openssl@3/3.4.0: 7,236 files, 33.4MB
```

Додаємо змінні середовища

```
❯ export PATH="/usr/local/opt/openssl/bin:$PATH"
❯ export LDFLAGS="-L/usr/local/opt/openssl/lib"
❯ xport CPPFLAGS="-I/usr/local/opt/openssl/include"
zsh: command not found: xport
❯ export CPPFLAGS="-I/usr/local/opt/openssl/include"
❯
❯
❯ export CPPFLAGS="-I/usr/local/opt/openssl/include"
```

Перевіримо встановлення

```
❯ openssl version
OpenSSL 3.4.0 22 Oct 2024 (Library: OpenSSL 3.4.0 22 Oct 2024)
```

Протестуємо
Шифрування та дешифрування
aes-256-cbc

```
❯ openssl speed aes-256-cbc
Doing aes-256-cbc ops for 3s on 16 size blocks: 122627221 aes-256-cbc ops in 3.00s
Doing aes-256-cbc ops for 3s on 64 size blocks: 49518051 aes-256-cbc ops in 2.99s
Doing aes-256-cbc ops for 3s on 256 size blocks: 12997782 aes-256-cbc ops in 3.00s
Doing aes-256-cbc ops for 3s on 1024 size blocks: 3291909 aes-256-cbc ops in 3.00s
Doing aes-256-cbc ops for 3s on 8192 size blocks: 414598 aes-256-cbc ops in 2.99s
Doing aes-256-cbc ops for 3s on 16384 size blocks: 207326 aes-256-cbc ops in 3.00s
version: 3.4.0
built on: Tue Oct 22 12:26:59 2024 UTC
options: bn(64,64)
compiler: clang -fPIC -arch arm64 -O3 -Wall -DL_ENDIAN -DOPENSSL_PIC -D_REENTRANT -DOPENSSL_BUILDING_OPENSSL -DNDEBU
G
CPUINFO: OPENSSL_armcap=0x987d
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
aes-256-cbc     654011.85k  1059918.15k  1109144.06k  1123638.27k  1135915.32k  1132276.39k
                                                                                18s 18:25:06
```

Хеш-функція
sha256

```
❯ openssl speed sha256
Doing sha256 ops for 3s on 16 size blocks: 26924433 sha256 ops in 3.00s
Doing sha256 ops for 3s on 64 size blocks: 24017314 sha256 ops in 3.00s
Doing sha256 ops for 3s on 256 size blocks: 15558772 sha256 ops in 2.99s
Doing sha256 ops for 3s on 1024 size blocks: 5952778 sha256 ops in 3.00s
Doing sha256 ops for 3s on 8192 size blocks: 878592 sha256 ops in 3.00s
Doing sha256 ops for 3s on 16384 size blocks: 445590 sha256 ops in 2.99s
version: 3.4.0
built on: Tue Oct 22 12:26:59 2024 UTC
options: bn(64,64)
compiler: clang -fPIC -arch arm64 -O3 -Wall -DL_ENDIAN -DOPENSSL_PIC -D_REENTRANT -DOPENSSL_BUILDING
G
CPUINFO: OPENSSL_armcap=0x987d
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
sha256          143596.98k   512369.37k  1332122.28k  2031881.56k  2399141.89k  2441654.37k
```

Crypto++

Встановлюємо за допомогою homebrew

```
❯ brew install cryptopp
==> Downloading https://formulae.brew.sh/api/formula.jws.json
######################################################################################################################## 100.0%
==> Downloading https://formulae.brew.sh/api/cask.jws.json
######################################################################################################################## 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/cryptopp/manifests/8.9.0-1
######################################################################################################################## 100.0%
==> Fetching cryptopp
==> Downloading https://ghcr.io/v2/homebrew/core/cryptopp/blobs/sha256:d6cdbe84008d6489b21fed34813dbc6b349c90c52cc75
######################################################################################################################## 100.0%
==> Pouring cryptopp--8.9.0.arm64_sequoia.bottle.1.tar.gz
🍺  /opt/homebrew/Cellar/cryptopp/8.9.0: 201 files, 13.9MB
==> Running `brew cleanup cryptopp`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
~ ..... 6s 18:35:55
❯
```

Перевіримо встановлення

```
❯ brew info cryptopp
==> cryptopp: stable 8.9.0 (bottled), HEAD
Free C++ class library of cryptographic schemes
https://cryptopp.com/
Installed
/opt/homebrew/Cellar/cryptopp/8.9.0 (201 files, 13.9MB) *
  Poured from bottle using the formulae.brew.sh API on 2024-11-14 at 18:35:55
From: https://github.com/Homebrew/homebrew-core/blob/HEAD/Formula/c/cryptopp.rb
License: LicenseRef-Homebrew-public-domain AND BSL-1.0
==> Options
--HEAD
        Install HEAD version
==> Analytics
install: 78 (30 days), 235 (90 days), 938 (365 days)
install-on-request: 71 (30 days), 187 (90 days), 804 (365 days)
build-error: 0 (30 days)
~ .....
❯
```

код для тестування

Users > a1 > crypto.cpp

```cpp
#include <cryptopp/aes.h>
#include <cryptopp/hex.h>
#include <cryptopp/filters.h>
#include <cryptopp/osrng.h>
#include <iostream>

using namespace CryptoPP;

int main() {
    AutoSeededRandomPool prng;
    byte key[AES::DEFAULT_KEYLENGTH];
    prng.GenerateBlock(key, sizeof(key));

    std::string plain = "Hello, Crypto++!";
    std::string cipher, encoded;

    try {
        // Налаштування шифрування
        ECB_Mode< AES >::Encryption e;
        e.SetKey(key, sizeof(key));

        // Шифрування тексту
        StringSource ss1(plain, true,
            new StreamTransformationFilter(e,
                new StringSink(cipher)
            )
        );

        // Перетворення зашифрованого тексту в hex-формат для відображення
        StringSource ss2(cipher, true,
            new HexEncoder(
                new StringSink(encoded)
            )
        );

        std::cout << "Cipher text: " << encoded << std::endl;
    } catch(const CryptoPP::Exception& e) {
        std::cerr << e.what() << std::endl;
        return 1;
```

перейдемо в дерикторію з файлом

```
❯ cd /Users/a1
❯ ls
Aforlabs.pem                    crypto.cpp
Applications                    dict.json
```

та скомпілюємо його

```
❯ g++ crypto.cpp -o crypto_test -lcryptopp
crypto.cpp:1:10: fatal error: 'cryptopp/aes.h' file not found
#include <cryptopp/aes.h>
         ^~~~~~~~~~~~~~~~~
1 error generated.
❯ g++ crypto.cpp -o crypto_test -I/usr/local/include -L/usr/local/lib -lcryptopp
crypto.cpp:1:10: fatal error: 'cryptopp/aes.h' file not found
#include <cryptopp/aes.h>
         ^~~~~~~~~~~~~~~~~
1 error generated.
```

при спробі компіляції бачимо помилку
вкажемо шляхи до заголовочних файлів та бібліотек явно

```
❯ g++ crypto.cpp -o crypto_test -I/opt/homebrew/Cellar/cryptopp/8.9.0/include -L/opt/homebrew/Cellar/cryptopp/8.9.0/
lib -lcryptopp
                                                                                                    18:56:21
```

компіляція виконано успішно

Запустимо програму та отримаємо зашифрований текст

```
❯ ./crypto_test
Cipher text: 2F81DFAE784D6ACEDA63603DF82E47D9B3B519C5B536C9007E3AAD0C15A676EF
```

Тепер протестуємо продуктивність для цього додамо оцінки часу виконання шифрування дешифрування та хешування в код

```
12  void testAES_CBC() {
19      std::string plain = "This is a test message for AES-256-CBC performance measurement!";
20      std::string cipher, recovered;
21
22      // Налаштування шифрування
23      CBC_Mode< AES >::Encryption encryption;
24      encryption.SetKeyWithIV(key, sizeof(key), iv);
25
26      // Налаштування дешифрування
27      CBC_Mode< AES >::Decryption decryption;
28      decryption.SetKeyWithIV(key, sizeof(key), iv);
29
30      // Початок вимірювання часу для шифрування
31      auto start = std::chrono::high_resolution_clock::now();
32      for (int i = 0; i < 1000; ++i) {
33          cipher.clear();
34          StringSource(plain, true,
35              new StreamTransformationFilter(encryption,
36                  new StringSink(cipher)
37              )
38          );
39      }
40      auto end = std::chrono::high_resolution_clock::now();
41      std::chrono::duration<double> elapsed = end - start;
42      std::cout << "AES-256-CBC encryption time (1000 iterations): " << elapsed.count() << " seconds" <
43
44      // Початок вимірювання часу для дешифрування
45      start = std::chrono::high_resolution_clock::now();
46      for (int i = 0; i < 1000; ++i) {
47          recovered.clear();
48          StringSource(cipher, true,
49              new StreamTransformationFilter(decryption,
50                  new StringSink(recovered)
51              )
52          );
53      }
54      end = std::chrono::high_resolution_clock::now();
55      elapsed = end - start;
56      std::cout << "AES-256-CBC decryption time (1000 iterations): " << elapsed.count() << " seconds" <
```

компілюємо

```
❯ g++ crypto.cpp -o crypto_test -I/opt/homebrew/Cellar/cryptopp/8.9.0/include -L/opt/homebrew/Cellar/cryptopp/8.9.0
lib -lcryptopp
❯ ./crypto_test
Cipher text: 2F81DFAE784D6ACEDA63603DF82E47D9B3B519C5B536C9007E3AAD0C15A676EF
❯ g++ crypto.cpp -o crypto_perf -I/opt/homebrew/Cellar/cryptopp/8.9.0/include -L/opt/homebrew/Cellar/cryptopp/8.9.0
lib -lcryptopp
crypto.cpp:31:5: warning: 'auto' type specifier is a C++11 extension [-Wc++11-extensions]
    auto start = std::chrono::high_resolution_clock::now();
    ^
crypto.cpp:40:5: warning: 'auto' type specifier is a C++11 extension [-Wc++11-extensions]
    auto end = std::chrono::high_resolution_clock::now();
    ^
crypto.cpp:64:5: warning: 'auto' type specifier is a C++11 extension [-Wc++11-extensions]
    auto start = std::chrono::high_resolution_clock::now();
    ^
crypto.cpp:74:5: warning: 'auto' type specifier is a C++11 extension [-Wc++11-extensions]
    auto end = std::chrono::high_resolution_clock::now();
    ^
4 warnings generated.
```

Запускаємо

```
❯ ./crypto_perf
Testing AES-256-CBC Performance:
AES-256-CBC encryption time (1000 iterations): 0.00555383 seconds
AES-256-CBC decryption time (1000 iterations): 0.00479896 seconds

Testing SHA-256 Performance:
SHA-256 hashing time (1000 iterations): 0.00150667 seconds
~
❯
```

PyCrypto

встановлюємо бібліотеку PyCryptodome (актуальна версія PyCrypto)

```
❯ pip3 install pycryptodome
Collecting pycryptodome
  Downloading pycryptodome-3.21.0-cp36-abi3-macosx_10_9_universal2.whl.metadata (3.4 kB)
Downloading pycryptodome-3.21.0-cp36-abi3-macosx_10_9_universal2.whl (2.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.5/2.5 MB 10.0 MB/s eta 0:00:00
Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.21.0

[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: pip3 install --upgrade pip
~                                                                                19:10:59
❯
```

Напишемо код на python для тестування

```python
from Crypto.Cipher import AES
from Crypto.Hash import SHA256
from Crypto.Random import get_random_bytes
import time

# Функція для тестування AES-256-CBC
def test_aes_cbc():
    key = get_random_bytes(32)
    iv = get_random_bytes(16)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    plaintext = b'This is a test message for AES-256-CBC performance measurement!' * 2

    # Шифрування
    start_time = time.time()
    for _ in range(1000):
        ciphertext = cipher.encrypt(plaintext.ljust(64))
    end_time = time.time()
    print("AES-256-CBC encryption time (1000 iterations):", end_time - start_time, "seconds")

    # Дешифрування
    cipher = AES.new(key, AES.MODE_CBC, iv)
    start_time = time.time()
    for _ in range(1000):
        decrypted_text = cipher.decrypt(ciphertext)
    end_time = time.time()
    print("AES-256-CBC decryption time (1000 iterations):", end_time - start_time, "seconds")

# Функція для тестування SHA-256
def test_sha256():
    message = b'This is a test message for SHA-256 performance measurement!'
    start_time = time.time()
    for _ in range(1000):
        hasher = SHA256.new()
        hasher.update(message)
        digest = hasher.digest()
    end_time = time.time()
    print("SHA-256 hashing time (1000 iterations):", end_time - start_time, "seconds")
```

Запустимо його

```
❯ python3 crypto.py
Testing AES-256-CBC Performance:
AES-256-CBC encryption time (1000 iterations): 0.0014450550079345703 seconds
AES-256-CBC decryption time (1000 iterations): 0.0014650821685791016 seconds

Testing SHA-256 Performance:
SHA-256 hashing time (1000 iterations): 0.003442049026489258 seconds
~
```

Висновки: в даній лабораторній роботі ми порівняли бібліотеки: OpenSSL, Crypto++, PyCryptodome для створення криптосистеми на ОС MacOS.
Наглядно можемо побачити що бібліотека PyCryptodome працює швидше Crypto++ та значно швидше ніж OpenSSL. Проте слід зазначити, що OpenSSL для використання значно простіший!